An Ontology Design Pattern for Cooking Recipes – Classroom Created

Monica Sam, Adila Alfa Krisnadhi, Cong Wang, John Gallagher, Pascal Hitzler

Department of Computer Science and Engineering, Wright State University, USA

Abstract. We present a description and result of an ontology modeling process taken to the classroom. The application domain considered was cooking recipes. The modeling goal was to bridge heterogeneity accross representational choices by developing a content ontology design pattern which is general enough to allow for the integration of information from differing web sites. We will discuss the pattern developed, and report on corresponding insights and lessons learned.

1 Introduction

The pattern which we will describe in this paper was developed as part of a class which the last named author conducted at Wright State University in Spring 2014. The class was called *Knowledge Representation for the Semantic Web*, and was an entry-level graduate class, with the goal of conveying to the students the fundamentals of Semantic Web knowledge representation languages.

The teacher had been experimenting in the past with a rather "classical" approach to teaching this subject, which essentially followed [6]. In this approach, topics progressed from light-weight to heavy-weight, starting with RDF and RDFS, their semantics and completion algorithms, the OWL syntax and intuitive semantics, followed by the introduction of description logics, their formal semantics and tableaux-based reasoning procedures. Practical modeling exercises progressed from shallow to deep, essentially starting with taxonomies which were then step by step (as material in the class progressed) expanded and adorned with more sophisticated axiomatization. Logical foundations were conveyed without proofs, as in the textbook, with the goal of allowing the students to develop a rough intuition of the logical underpinnings without diving too deeply into these issues.

While the classes (and the textbook, for that matter) were well received by the students, these classes fell short of one particular goal which the teacher had had: most students did *not* develop an intuition for formal semantics and logical deduction, i.e. the understanding of the languages remained at an informal and rather shallow and mostly syntactic level. While this is not necessarily a problem regarding useful learning outcomes for the students, the teacher was motivated to pursue alternative routes.

For the class in Spring 2014, contents and approach were radically modified. Explicit treatment of RDF and OWL was deferred to a brief overview at the end

of the class, while the bulk of the material taught was on formal logic around description logics and rules (datalog and logic programming), plus some existential rules, with full formal proofs for undecidability results and for correctness of the presented algorithms. The main emphasis was thus to engulf the students in the formal matter, at which point the concrete W3C standards like RDF and OWL became almost afterthoughts, the syntax and semantics of which could be conveyed very quickly in the end as special cases of what had been dealt with on a more foundational level over the course of four months.

A second major change was adopted: The former class project, which was to built an ontology progressively starting from light-weight (taxonomy) to heavy-weight (OWL and rules axioms) was completely dropped. Instead, the whole class met for a 7-hour session (with breaks) on a Saturday to practice collaborative ontology design pattern (ODP) modeling. In the two regular class sessions before that Saturday the students had received a brief introduction to patterns and the patterns idea, as well as a minimalistic primer on RDF and linked data. The primary examples discussed were the Semantic Trajectory pattern [7], the Cruise pattern [9] and from Linked MDB (linkedmdb.org). The students did know that they would collaboratively model a content ODP, but they did not learn before the class session what notion it would be.

For the modeling session, the class was split into two groups, which initially were to work independently. The charge was as follows.

Design an ontology design pattern which can be used as part of a "recipe discovery" website. The pattern shall be set up such that content from existing recipe websites can "in principle" be mapped to the pattern (i.e., the pattern gets populated with data from the recipe websites). On the discovery website, detailed graph-queries (using the pattern) shall produce recipes from different recipe websites as results.

Students were instructed to first look at some popular recipe websites, then to formulate, in natural language, some example queries to the website such as "I have cabbage and potatoes, how can I make that into a nice meal in 45 minutes?" They were then to develop a graph structure as basis for a pattern and check that their queries work with these.

In a second stage, the groups were mixed, with one representative from each group changing to the other group. This representative brought the original questions from his original group and both patterns were modified to accomodate the additional types of queries. The groups then added axioms in the form of OWL or rules to constrain the formal semantics. Figures 1 and 2 depict the two patterns the groups came up with, and we will discuss these further in Section 4.

In a third stage, each group was charged with describing how their pattern could be mapped to the other pattern, and asked to report on the problem points, i.e., where and why mapping of parts of the pattern didn't really work.

The whole class finally merged all insights into one draft pattern, which is depicted in Figure 3 with only minor modifications – mostly regarding the choice of names for classes and properties, and a more detailed axiomatization. This resulting pattern is laid out in detail in this paper. To the teacher's own surprise,

the pattern turned out very well, and the class feedback was excellent. We will discuss further aspects of the class pattern modeling experience towards the end and discuss the resulting pattern.

2 Modeling Recipe

The term *recipe* has several contextual meanings. It can be defined in a general sense as a method to obtain a desired end. When used in the context of cooking, it is generally considered to be a set of instructions on preparing a culinary dish. As such, it could be viewed as an object with properties such as ingredients and time needed. Alternatively, it could be viewed as a *process*, which takes in some input, has a series of steps to be executed, and produces some output. The time taken to execute the steps and the utensils needed also help describe the recipe.

A cooking recipe, apart from the instructions and ingredients, sometimes contain information that help categorize it better for various needs. A recipe could be described as vegetarian or belonging to a particular course and appropriately categorized. Application-specific information may also be needed to tailor the recipe for specific purposes.

Modeling a piece of particular domain knowledge as an ODP typically requires involvement of domain experts. In modeling recipes for this paper, however, the presence of domain experts is not strictly required since information about recipes is widely available online and easily understandable by most lay people. Moreover, we are not so much interested in the actual resulting pattern, but rather on the experience of using ODP as a pedagogical element for teaching Semantic Web knowledge representation languages. In place of sessions with domain experts, a brainstorming session in the class was carried out to gather the information necessary to model recipe in this paper. The resulting pattern was obtained from scratch since the participants were not exposed to previous attempts for modeling cooking recipe or, in general, food and the related notions, some of which we survey below.

Cantais et al. [2] presented a Food ontology, which works together with the Diets and Product ontologies to provide a recommendation to users on a healthy choice of food in accordance to their health condition. They proposed a formalization based on a use case scenario over diabetic patients. Recipe is not modeled explicitly in these ontologies, although the recommender system that uses them can suggest combination of ingredients appropriate to the dieting requirements.

Mota and Agudo [10] presented an ontology of ingredients, which was used as a part of a recipe adaptation system. Recipe is not modeled as a process as in our pattern, but rather, as a simple collection of those ingredients that can be added when needed or removed when unwanted.

Ribeiro et al. [11] described a Cooking ontology as an example how a spoken dialogue system can be enriched with a domain knowledge. The ontology itself is very fine-grained and complex containing more than 1000 classes divided into seven ontology modules. Recipe is modeled here as a combination of preparation, cooking, and presentation phases, each of which consists of a sequence of tasks.

Other relevant notions such as ingredient, utensil, measure, and unit are also included to model various situations, which may involve variation of measures and units, classification of food items (e.g., alcoholic and non-alcoholic), and differences in courses of the food item due to geographical locations. Recipe can thus be seen as a process like in our pattern, although in our case, the pattern is not as fine-grained as this one to retain a relatively high degree of extensibility.

To populate our pattern (or suitable extension thereof) with data, one can in principle use an approach using indentation and tagging structures in XML documents to learn to extract recipe information in the form of semantic annotations from web pages [4].

3 Competency Questions

For the modeling exercise in class, the two groups were first asked to look at existing recipe websites and formulate the kind of questions the information in the websites helped answer. Group A made the assumption that recipe websites would contain information in standard data types and tried to drill down information to the most primitive level. Accordingly, their questions were also more specific. In the following section, we will look at some of the questions each group came up with, and how the two groups modeled similar information differently. Some of the questions that Group A came up with were as follows:

Question 1 "Breakfast dishes I can prepare with 2 potatoes and 100 grams of wheat flour."

Here, group A assumed that quantities such as 2 potatoes and 100 grams of wheat flour should be clearly and unambiguously presented in each website. Group A modeled this information using the FoodItem and quantityOfFood classes. Group B had included ingredient information in their Ingredient and Quantity classes.

Question 2 "Gluten-free desserts with less than 100 calories."

Group A modeled all food categorization in a single class called FoodClass. Food classifications such as vegetarian, gluten-free, flavor information such as spicy, sweet, foods with classifications such as Easter eggs, cuisine and course information is modeled by this class. The above question would be answered by Group A's model using the FoodClass and NutritionInfo classes, and may be answered by the Keyword class in Group B's model, if the recipe had the correct keyword associated with it and the NutritionInfo classes.

Question 3 "Mexican dishes which do not use chili."

An interesting aspect of the model of Group A is that both ingredients and food products are modeled as foods with quantities. This will be discussed later in detail. The above question would be answered by the FoodClass and isOptional property of the Ingredient class in Group A's model. Group B modeled this

information using their Cuisine class. It might have been more difficult to query for the absence of an ingredient using Group B's model.[1] Group A was certain that there should be a difficulty level associated with each recipe and included that in their model. They also associated an Author pattern with each of their recipes. This would help answer questions such as the following:

Question 4 "Easy Gordon Ramsey breakfast dishes."

It was noted that the problem of uniformity in representing levels of difficulty across websites might show up when modeling this concept. Whereas Group A had an Author associated with each recipe, Group B did not. Therefore the above question couldn't be answered by their model. Cooking utensils needed and time to cook were some information that was agreed to be vital to answer questions such as the following:

Question 5 "Grilled meat in less than 1 hour."

The above utensil and time information were modeled by both groups. Group A used the utensil and Time classes for this and Group B used CookingTool and OWL:Time classes for this. Group B also had some similar ideas as to what ought to be included in a model for recipe websites. They identified flavor, texture, cuisine and serving temperature as essential attributes of any recipe, to answer questions such as the following:

Question 6 "Spicy Korean beef dishes."

For the above question, Group B's Cusinie, Flavor and Ingredient classes were modeled. Group A would have answered this question with the FoodClass and Ingredient classes.

Question 7 "Crunchy brownie recipes."

Group B modeled the Texture class and Keyword class to answer queries such as the above, whereas Group A modeled this information using the FoodClass and Keyword classes.

Question 8 "Cold appetizers."

Group B had a specific ServingTemperature class to answer queries on the temperature the dish should be served at, while Group A modeled such information only with FoodClass and Keyword classes. They also included cooking utensil and difficulty level:

Question 9 "Baked/Mashed potatoes."

The above question would need information on the utensil to be used to prepare the recipe, since baking involves an oven and mashing presumably requires a masher or some similar utensil, and this information is modeled using the Utensil class by Group A and CookingTool class by Group B.

[1] (Local) closed world reasoning is of course also required for this type of query.

Name	Type	Explanation
hasIngredient	*Recipe* × *QuantityOfFood*	An ingredient of the recipe
produces	*Recipe* × *produces*	A dish produced by the recipe
consistsOf	*QuantityOfFood* × *FoodItem*	The quantity of either an ingredient or dish
classifiedAs	*FoodItem* × *FoodClassification*	The classification of an ingredient or disth
hasName	*FoodItem* × *String*	The name of an ingredient or dish
hasProcessDescription	*Recipe* × *ProcessDescription*	the description of the recipe process
hasTextualDescription	*ProcessDescription* × *Document*	The text description of the recipe process
preparationTime,cookingTime	*Recipe* × *timeInterval*	The duration of the recipe process
requires	*Recipe* × *Utensil*	A utensil needed for the recipe
hasName	*Recipe* × *String*	The name of a recipe
hasInformationObject	*Recipe* × *InformationObject*	The information object of a recipe
hasURL	*InformationObject* × *URL*	The URL information of a recipe
hasKeyword	*InformationObject* × *String*	The keyword information of a recipe
hasAuthor,hasRecommender	*InformationObject* × *Person*	The author information of a recipe
hasDifficultyLevel	*InformationObject* × *DifficultyLevel*	The difficulty level information of a recipe
hasRating	*InformationObject* × *Rating*	The rating of a recipe

Table 1: Basic relations needed to answer the competency questions.

Question 10 "Easy desserts with less than 100 calories."

Both the groups had modeled a DifficultyLevel class and NutritionalInfo class. The above query would be answered by these classes. Some information such as nutritional information, time to cook, utensils needed, difficulty level were identified and modeled similarly by both groups. In order to answer these questions, an ontology design pattern needs to distinguish a number of relations. We introduce these abstract relations in Table 1, before formalizing them in Section 5.

4 In-class Modeling – A Discussion

Based on these competency questions, the two groups of students came up with two rather different drafts of a recipe pattern. These drafts are depicted in Figures 1 (Group A) and 2 (Group B).

Initially, both the groups modeled recipe as a class with relationships to other classes. So they defined properties that objects of the class would have. There were ingredients which became a property of the class and there were instructions that was also a property of the class, as was the dish that was produced. With this approach instructions were just another property of the recipe. An alternate way was proposed by the teacher, in which the recipe became a process, and therefore a description of the procedure became now a defining characteristic of it. The students then modified their patterns based on this thinking.

While Group A connected both ingredients and food products as belonging to a single category, which they called FoodItem, Group B continued to view them separately and modeled them as Product and Ingredient. Both the groups

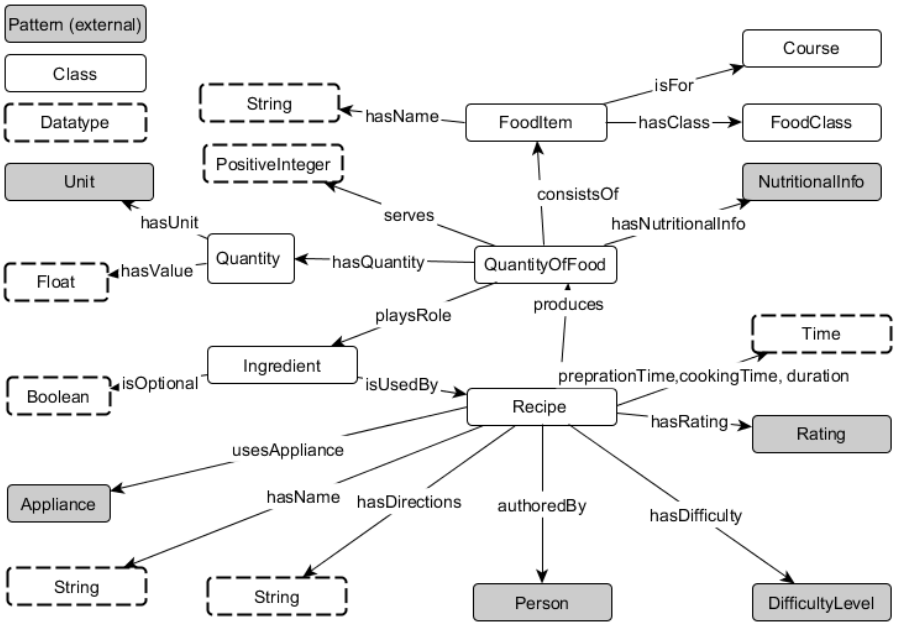

Fig. 1: Recipe modeled by group A.

independently came to the conclusion that there had to be some modeling of the concept of quantity to be associated with FoodItems. Group A decided that quanity would in turn have a unit and a value associated with it, whereas Group B did not make that decision. Group A associated all the information related to categorization – based on spice levels, based on cuisine, based on dietary content as all belonging to a single class they called FoodClass. They had a separate class for the course of the dish the recipe was for. Group B decided to maintain separate, the notions of Cusine, Flavor and Texture. Both the groups independently came up with the idea of nutritionalInfo that had nutritional information about the foodItem produced as a result of the recipe. Both the groups came up with ideas of time associated with preparation and cooking and the appliances used to make the food item. Both the groups also came up with the idea of a difficulty level associated with the procedure. While the ingredient had an associated property which marked whether it was optional in the design of Group A, Group B had not modeled that information. Also Group A had an author associated with the recipe which Group B did not, while Group B had a Photo associated with the recipe that Group A did not. Group A also had a Rating associated with the recipe that Group B did not.

Of course, the two groups made different modeling choices, and as part of the class session, after producing the initial drafts just discussed, the two groups were asked to attempt to develop mappings, one group from version A to version B,

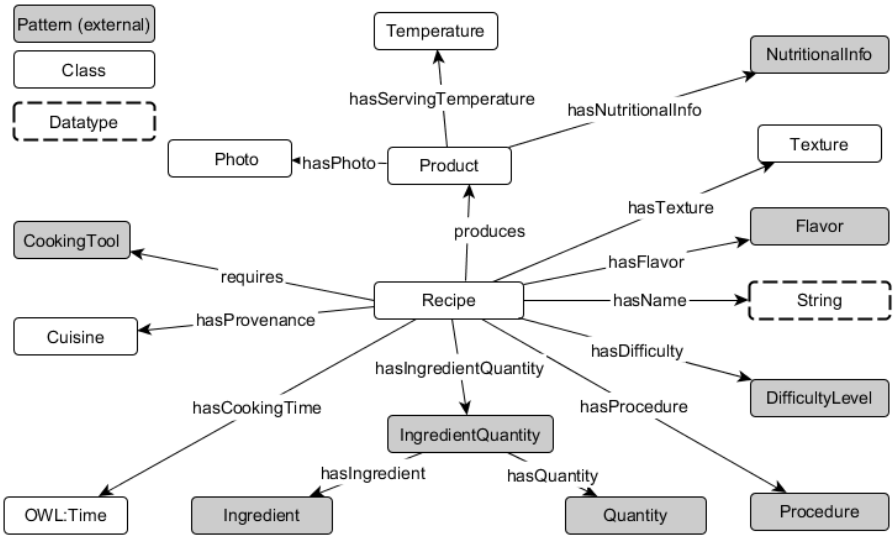

Fig. 2: Recipe modeled by group B.

and one group vice versa as the third stage of the class modeling session (see the overview in Section 1). They were charged with developing a loss-less mapping, if possible, and to report on issues where such a mapping was not possible. This task aligned with the general charge for the modling session (see Section 1): the charge called for a very general pattern to which content could always be mapped. So difficulties found in the mapping exercise would indicate too specific modeling choices.

A direct mapping could indeed be made between the two models for the following concepts pairs (and corresponding properties): Recipe – Recipe; Ingredient as QuantityOfFood – Ingredient with IngredientQuantity; Quantity – Quantity; Appliance – CookingTool; DifficultyLevel – DifficultyLevel; Name – Name; Directions – Procedure; NutritionalInfo – NutritionalInfo; Time – OWL:Time; FoodClass – Cuisine.

The individual pieces of information about the recipe that each model had which was not in the other model could not be accomodated unless the model was expanded. The author modeled in Group A's version, the photograph modeled in Group B's version could not be mapped directly. The unit and value associated with Quanitity in Group A's model would need an expansion of Group B's model. The Temperature modeled in Group B's model could not be accomodated in Group A's model. The Optional Concept associated with Group A's model could not be directly mapped into Group B's model. The Rating associated with Group A's model had no equivalent in Group B's model. The Course associated with Group A's model had no equivalent in Group B's model. The choice of Group A to use a class QuantityOfFood which can act both as recipe ingredient

and as recipe product found general agremment, partially because of the perceived conceptual clarity of the modeling, partially because it was realized that sometimes the recipe product becomes in turn an ingredient in another recipe. So the apporach by Group A was adopted for the final version.

The mapping attempts furthermore exposed strong ontological commitments regarding some classes used as the range for properties, e.g. demanding a Float as quantity value. In the ensuing discussion it was realized that the quantity value class may be a complex entity, i.e. a pattern in its own right. E.g., recipes may specify a "pinch of salt", or "salt and pepper to taste".

Missing information about Photograph in Group A's model and Author, Rating in Group B's model was filled by creating a new class called InformationObject which conceptualized information associated with the Recipe. The missing information in Group A's model about serving temperature, flavor and cuisine and in Group B's model about course and food class were modeled into another object called FoodClass. By associating a quanity with ingredient, it was decided that the need for an optional indicator field was removed.

The in-class discussion, in particular seeing the different modeling choices, the attempt and failure to produce mappings, and the final consensus to establish the final pattern draft (Figure 3) provided the students with an understanding of the difficulties of making, using, and reusing conceptual models, which would have been much more difficult to convey without a group modeling exercise.

5 Finalized Pattern and OWL Formalization

We now present the finalized design pattern, based on the previously described conceptual foundations. In the following, we respectively discuss the classes and properties within the pattern and formally encode them using the Web Ontology Language (OWL) [5]. We make use of Description Logics (DL) [6] notation, as we believe this improves the readability and understandability of the axioms presented. Note that tractable reasoning is important for producing an efficient implementation of the pattern. A schematic view of the pattern is shown in Figure 3.

Recipe. A Recipe is a class which is described as a process. It may take as ingredients instances of QuantityOfFood and also produce instances of QuantityOfFood: Recipe $\sqsubseteq \exists$hasIngredient.QuantityOfFood$\sqcap\exists$produces.QuantityOfFood A recipe also requires some utensils for execution which is modeled as a Utensil pattern, and requires some amount of time for execution described by a pattern named TimeInterval. Each instance of the recipe class also has a name which is of the String data type. Information about the recipe is present in a class called InformationObject.

QuantityOfFood. QuantityOfFood is a class that models the quantity of both ingredients and food products produced by the recipe. It can be described as serving a number of people which is of datatype positive integer. It also has some

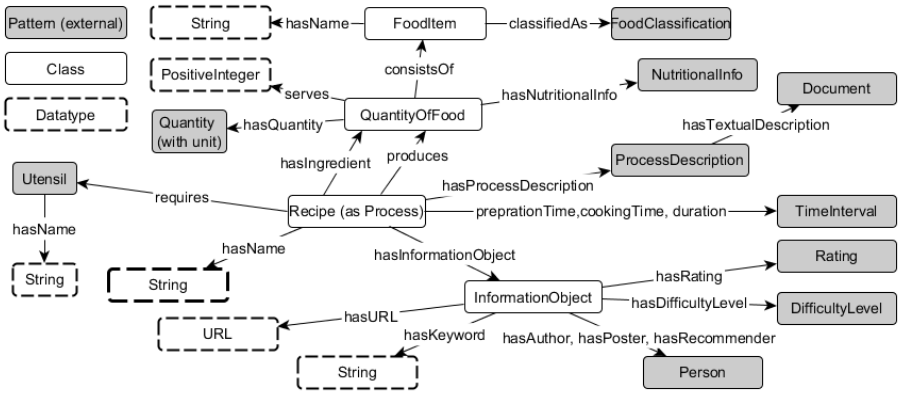

Fig. 3: Recipe as a process modeled along with other patterns (marked grey) needed to describe a recipe.

nutritional information associated with it, which is modeled as a pattern called nutritionalInfo. The quantity it has is described using a pattern called quantity which is some quantity along with a unit: $\exists \text{produces}^{-}.\text{Recipe} \sqsubseteq \text{QuantityOfFood}$

FoodItem. FoodItem is a class that models an ingredient or food product which in turn has its quantity described by QuantityOfFood. The FoodItem has a name which is modeled using a String datatype and has a food classification which in turn is defined as a pattern named FoodClassification. FoodClassification has been modeled in a very broad sense to include such classification as vegetarian, vegan, gluten-free to seasonal, or occassion-specific information such as thanksgiving meals or even meals for a purpose such as lunch-box meals: $\text{QuantityOfFood} \sqsubseteq =1\text{consistsOf.FoodItem} \sqcap =1\text{hasQuantity.Quantity}$

InformationObject. InformationObject is a class that contains information needed to describe a recipe. Some of its properties are hasURL, hasKeyword (of String datatype), hasAuthor, hasPoster, hasRecommender - which is a Person described by an external pattern, and a difficulty level, also modeled as an external pattern. It also includes Rating information, which also could pose challenges, if the rating systems are all not normalized.

OWL Profile. Additional axioms for the pattern are given in Appendix A. From these axioms, we know our recipe pattern falls into the Description Logic $\mathcal{ELIF}(D)$. An $\mathcal{ELIF}$-Tbox can be reduced to an $\mathcal{ELI}$-Tbox whose size is linear in the size of the original one [12]. In $\mathcal{ELI}$, subsumption w.r.t. GCIs is ExpTime-complete [1], and $\mathcal{ELI}$ knowledge bases can be classified by a completion-rule based algorithm[2] [8,12]. However, if we rewrite range restrictions of properties

[2] A practical reasoner for this fragment, called CB reasoner, can be found at https://code.google.com/p/cb-reasoner/.

to be of universal scope, e.g., $\exists \text{hasIngredient}^{-}.\top \sqsubseteq \text{QuantityOfFood}$, $\mathcal{ELI}$ can even reduced to $\mathcal{EL}$ [3], such that the subsumption checking can be done in polynomial time [1].

6 Informal Evaluation

In this section, we illustrate how our pattern can be specialised to the content of specific websites.

For the website *www.allrecipes.com*, most of the information given for recipes can be modeled based on our pattern, except for the addition of a couple of InformationObjects of type image and video. The recipes also have a Review associated with them, but do not contain information on difficulty level.

Another example is the website *www.bettycrocker.com*. The information on this website can be modeled by a slight extension of our pattern to include expert tips to the process description. Also, this website has review information, and no difficulty level.

A third example is taken from *www.epicurious.com*. This website uses information such as the main ingredients, the dietary considerations this recipe would meet – for example, being vegan or high fiber and the season associated with the dish to tag the recipe. Recipes are also categorized. The information in this website could be modeled by of course adding the Review InformationObject, removing the difficultyLevel and adding in another InformationObject for popularity which lists the number of times the recipe had been downloaded. In all cases, the key page contents can be captured with specialzations of our recipe pattern. Further details are provided in Appendix B.

7 Conclusions

The class exercise produced a reasonable outcome in terms of the resulting content pattern. The students experienced the power of collaborative modeling to obtain versatile patterns, and overall the experience was resulted in very positive feedback. A lesson learned (for the teacher) is that it seems necessary to convey some amount of thorough logical underpinnings in order to effectively teach introductory ontology modeling.

Acknowledgements This work was partially supported by the National Science Foundation under award 1017225 "III: Small: TROn – Tractable Reasoning with Ontologies."

References

1. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: Kaelbling, L.P., Saffiotti, A. (eds.) Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30-August 5, 2005. pp. 364–369. Professional Book Center (2005)

2. Cantais, J., Dominguez, D., Gigante, V., Laera, L., Tamma, V.: An example of food ontology for diabetes control. In: Proceedings of the International Semantic Web Conference 2005 workshop on Ontology Patterns for the Semantic Web (2005)
3. Carral, D., Feier, C., Grau, B.C., Hitzler, P., Horrocks, I.: EL-ifying ontologies. In: Demri, S., Kapur, D., Weidenbach, C. (eds.) Automated Reasoning – 7th International Joint Conference, International Joint Conference on Aritificial Reasoning 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8562, pp. 464–479. Springer (2014)
4. Ciancarini, P., Iorio, A.D., Nuzzolese, A.G., Peroni, S., Vitali, F.: Semantic annotation of scholarly documents and citations. In: Baldoni, M., Baroglio, C., Boella, G., Micalizio, R. (eds.) AI*IA 2013: Advances in Artificial Intelligence – XIIIth International Conference of the Italian Association for Artificial Intelligence, Turin, Italy, December 4-6, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8249. Springer (2013)
5. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation 27 October 2009 (2009), available from http://www.w3.org/TR/owl2-primer/
6. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. CRC Press (2010)
7. Hu, Y., Janowicz, K., Carral, D., Scheider, S., Kuhn, W., Berg-Cross, G., Hitzler, P., Dean, M., Kolas, D.: A geo-ontology design pattern for semantic trajectories. In: Tenbrink, T., Stell, J.G., Galton, A., Wood, Z. (eds.) Spatial Information Theory – 11th International Conference, Conference on Spatial Information Theory 2013, Scarborough, UK, September 2-6, 2013. Proceedings. Lecture Notes in Computer Science, vol. 8116, pp. 438–456. Springer, Heidelberg (2013)
8. Kazakov, Y.: Consequence-driven reasoning for Horn SHIQ ontologies. In: Boutilier, C. (ed.) Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009. pp. 2040–2045 (2009)
9. Krisnadhi, A., Arko, R., Carbotte, S., Chandler, C., Cheatham, M., Finin, T., Hitzler, P., Janowicz, K., Narock, T., Raymond, L., Shepherd, A., Wiebe, P.: An ontology pattern for oceanographic cruises: Towards an oceanographer's dream of integrated knowledge discovery. OceanLink Technical Report 2014.1, available from http://www.oceanlink.org/ (2014)
10. Mota, S.G., Agudo, B.D.: ACook: Recipe adaptation using ontologies, case-based reasoning systems and knowledge discovery. In: Cordier, A., Nauer, E. (eds.) Proceedings of the Cooking With Computers workshop. pp. 41–45 (2012)
11. Ribeiro, R., Batista, F., Pardal, J.P., Mamede, N.J., Pinto, H.S.: Cooking an ontology. In: Aritificial Intelligence : Methodology, Systems, Applications. pp. 213–221. Springer (2006)
12. Vu, Q.H.: Subsumption in the Description Logic $\mathcal{ELHIFR}+$ w.r.t. General TBoxes. Ph.D. thesis, Technical University Dresden (2008)

A Additional Axiomatization

We present the complete axiomatization of the Recipe pattern in the following, together with some brief explanations. The set of axioms can be encoded in OWL DL.

First, every Recipe has some ingredient that is some food item with some quantity, modeled as QuantityOfFood.

$$\text{Recipe} \sqsubseteq \exists \text{hasIngredient.QuantityOfFood} \tag{1}$$

Every Recipe produces some food item with some quantity, likewise modeled as QuantityOfFood.

$$\text{Recipe} \sqsubseteq \exists \text{produces.QuantityOfFood} \tag{2}$$

Every Recipe has a ProcessDescription associated with it.

$$\text{Recipe} \sqsubseteq \exists \text{hasProcessDescription.ProcessDescription} \tag{3}$$

Every Recipe has some PreparationTime modeled as a TimeInterval.

$$\text{Recipe} \sqsubseteq \exists \text{hasPrepationTime.TimeInterval} \tag{4}$$

Every Recipe has some CookingTime modeled as a TimeInterval.

$$\text{Recipe} \sqsubseteq \exists \text{hasCookingTime.TimeInterval} \tag{5}$$

Every Recipe has some Duration modeled as a TimeInterval.

$$\text{Recipe} \sqsubseteq \exists \text{hasDuration.TimeInterval} \tag{6}$$

Every Recipe has some InformationObject associated with it.

$$\text{Recipe} \sqsubseteq \exists \text{hasInformationObject.InformationObject} \tag{7}$$

Every Recipe has a name. The name is expressed using xsd:string datatype

$$\text{Recipe} \sqsubseteq \exists \text{hasName.xsd:string} \tag{8}$$

Every Recipe has some NutritionalInfo associated with it.

$$\text{QuantityOfFood} \sqsubseteq \exists \text{hasNutritionalInfo.NutritionalInfo} \tag{9}$$

Every Recipe serves some positive quantity of meals.

$$\text{QuantityOfFood} \sqsubseteq \exists \text{serves.xsd:positiveInteger} \tag{10}$$

Every QuantityOfFood has exactly one foodItem associated with it.

$$\text{QuantityOfFood} \sqsubseteq =1 \text{consistsOf.FoodItem} \tag{11}$$

Every QuantityOfFood has exactly one Quantity associated with it.

$$\text{QuantityOfFood} \sqsubseteq =1\text{hasQuantity.Quantity} \tag{12}$$

Every InformationObject has a URI.

$$\text{InformationObject} \sqsubseteq \exists\text{hasURL.xsd:anyURI} \tag{13}$$

Every FoodItem is classified using FoodClassification.

$$\text{FoodItem} \sqsubseteq \exists\text{classifiedAs.FoodClassification} \tag{14}$$

Every FoodItem has a name, typed as xsd:string.

$$\text{FoodItem} \sqsubseteq \exists\text{hasName.xsd:string} \tag{15}$$

Every Utensil has a name.

$$\text{Utensil} \sqsubseteq \exists\text{hasName.xsd:string} \tag{16}$$

Every ProcessDescription has a Document associated with it.

$$\text{ProcessDecription} \sqsubseteq \exists\text{hasTextualDescription.Document} \tag{17}$$

Range and Domain scopes in DL

In addition to the above axioms, we also assert domains and ranges for the properties in the pattern. Given a property P whose domain is A and range is B, the most commonly used domain and range restrictions are unguarded and can be expressed respectively as the axiom $\exists P.\top \sqsubseteq A$ and $\exists P^{-}.\top \sqsubseteq B$. We, however, use the guarded version of domain and range restrictions, which are respectively of the form $\exists P.B \sqsubseteq A$ and $\exists P^{-}.A \sqsubseteq B$. This lessens the ontological commitment we have to make in the pattern. For example, for the domain restriction $\exists P.B \sqsubseteq A$, the domain A is entailed on an instance only when it connects to an instance of B through the property P, unlike the unguarded version in which the first part of any pair of elements connected via P will always be asserted as belonging to A, regardless whether or not the other pair is an element of B.

$$\exists\text{hasIngredient}^{-}\text{.Recipe} \sqsubseteq \text{QuantityOfFood} \tag{18}$$

$$\exists\text{hasIngredient.QuantityOfFood} \sqsubseteq \text{Recipe} \tag{19}$$

$$\exists\text{produces}^{-}\text{.Recipe} \sqsubseteq \text{QuantityOfFood} \tag{20}$$

$$\exists\text{produces.QuantityOfFood} \sqsubseteq \text{Recipe} \tag{21}$$

$$\exists\text{consistsOf}^{-}\text{.QuantityOfFood} \sqsubseteq \text{FoodItem} \tag{22}$$

$$\exists\text{consistsOf.FoodItem} \sqsubseteq \text{QuantityOfFood} \tag{23}$$

$$\exists\text{consistsOf}^{-}\text{.QuantityOfFood} \sqsubseteq \text{FoodItem} \tag{24}$$

$$\exists\text{consistsOf.FoodItem} \sqsubseteq \text{QuantityOfFood} \tag{25}$$

$$\exists\text{hasInformationObject}^{-}\text{.Recipe} \sqsubseteq \text{InformationObject} \tag{26}$$

$$\exists\text{hasInformationObject.InformationObject} \sqsubseteq \text{Recipe} \tag{27}$$

B Informal Evaluation: Adjusted pattern

We give the specialized patterns for the different examples from Section 6. For *www.allrecepies.com*, this can be found in Figure 4. For *www.bettycrocker.com*, see Figure 5. For *www.epicurious.com*, see Figure 6.

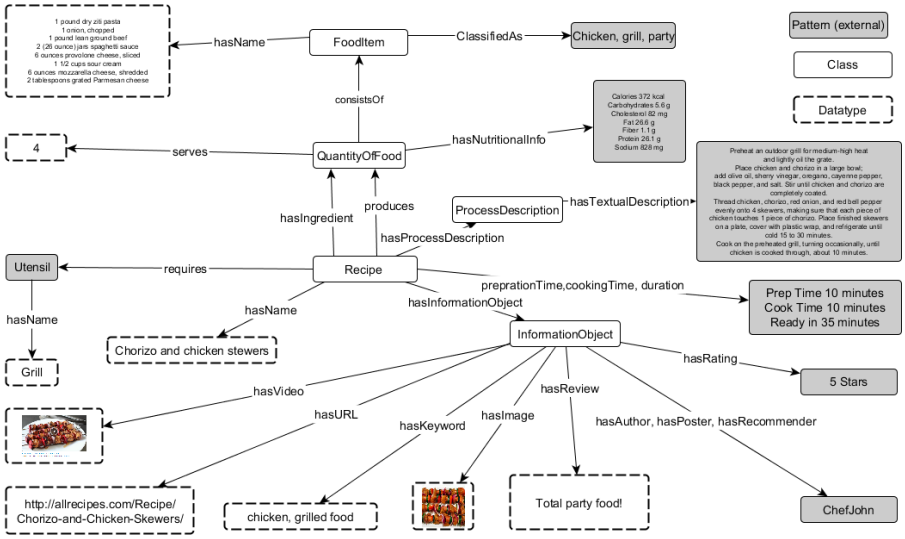

Fig. 4: The information in the AllRecipes website modeled.

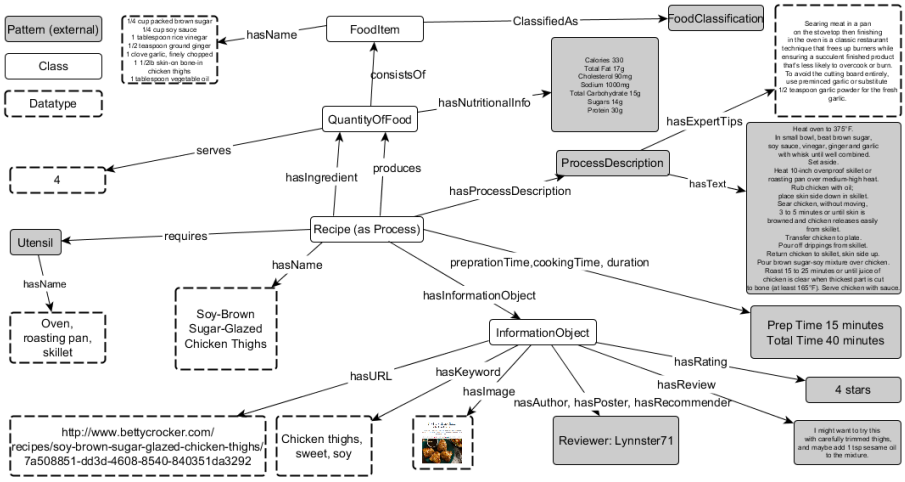

Fig. 5: The information in the Betty Crocker website modeled.

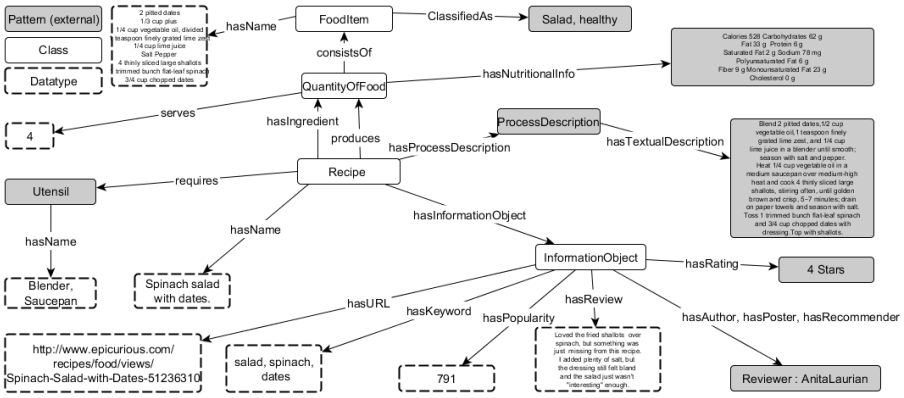

Fig. 6: The information in the Epicurious website modeled.